# ENSF 608 Final Project Database-- Design Decisions:

Zach Frena, Meet Pandya, Marko Mijovic

The **Users** entity is the main component to handle all the user-related relations as well as eventually helping with authorization. We chose to split up the **Users** relation into 5 specializations-- **Admins**, **Teaching Technician**, **Student**, **Care Attendants** and **Health Technicians**. This was done to make the database future proof if/when new user roles get added to the system. To handle authorization, we chose to represent access privileges as 2 attributes: uRole and accessLevel. uRole is a string attribute that contains values like ‘admin’, ‘teaching_technician’, ‘student’, and so on and so forth. The accessLevel attribute simply acts as a numerical coding schema to identify the user role with the schema rules being chosen by us (AccessLevel 5 is Admin, level 4 is teaching_technician, etc.).

An Admin can add and edit users, which we modeled as **Controls** (a strong relationship) inside the EER diagram. A teaching technician can borrow an animal, which we modeled as a strong **Borrows** relationship between the teaching technician and the animal entity. These strong relationships use foreign keys that reference another entity. Due to cardinality, a desire to not duplicate entries, and to maintain record of past entries, we added those relationships as new separate entities, **Logs** and **Borrows**.

The **Animal** entity is the main component of our application, as this is the entity that we are wanting to track and interact with in almost every use-case we perform. The Animal entity itself was designed to be standalone and not rely on any foreign keys. Its primary key (animalID) serves as foreign key in multiple relations also found in our database design, such as **Animal Health**, **Animal Diagnosis**, etc. The reason why **Animal Health** is stored as a separate entity within our database is because we wanted to preserve the treatment history of each animal, rather than just keeping a current snapshot of the animal’s current well-being. This design choice was a driving factor in our project, which may have resulted in a larger number of overall relations but was something we felt was necessary if more complex information collection was required in the future.

To best fulfill project requirements, we decided to model **Diagnosis**, **Treatment**, **Prescription**, and **Prescription Item** as independent strong entities. Diagnosis has a mandatory relationship with the animal relation, treatment has a mandatory relationship with diagnosis, and so on. This way we can effectively model 1-to-many relationships while not duplicating data entries, and maintaining a past record of previous entries.